Homework 1

CMSY-217, Fall 2011

The source code and sample output for this assignment must be submitted electronically using the Canvas course website prior to the start of class on Thursday, September 15.

1. Write a Java class called `QuickSort` which belongs to the package which is the reverse of your HCC email address. For example, my `QuickSort` class would belong to the package `edu.howardcc.mikemiller`.

2. Implement a recursive quicksort algorithm based on the following pseudocode:

```
SORT(A,p,r)
        if p < r then
                q := PARTITION(A,p,r)
                SORT(A,p,q)
                SORT(A,q+1,r)
```

```
PARTITION(A,p,r)
        x := A[p]
        i := p – 1
        j := r + 1
        while TRUE do
                repeat j := j – 1
                until A[j] ≤ x
                repeat i := i + 1
                until A[i] ≥ x
                if i < j then
                        exchange A[i] and A[j]
                else
                        return j
```

This initial method call to sort an array should be `QuickSort.sort(a,0,a.length-1);`

3. Make the `sort` and `partition` methods both `public` and `static`. The return type for the `sort` method is `void` while the `partition` method returns type `int`. Both methods take three parameters: a one-dimensional array of type `int`, a starting index of type `int`, and an ending index of type `int`.

4. Create integer arrays of various sizes and use the `nextInt` method from the `java.util.Random` class to initialize each element before sorting them with your `QuickSort.sort` method. Compare the performance of your quicksort algorithm to the one implemented in the Java 6 API by sorting the same size arrays with the `Arrays.sort` method from the `java.util` package. Record the runtimes in your table.

5. Demonstrate that the efficiency of your quicksort algorithm (on randomized data) is approximately $O(n \log n)$ by tabulating the runtimes for sorting different size arrays and plotting t versus $n \ln n$, where n is the size of the array and t is the runtime. Runtimes can be estimated by calling the static method `System.out.currentTimeMillis` immediately before and after the call to the `sort` method.

		QuickSort.sort	Arrays.sort
n	$n \ln n$	t	t'
1,000,000	1.382E+07	136	174
2,000,000	2.902E+07	282	359
4,000,000	6.081E+07	584	743
8,000,000	1.272E+08	1,220	1,530
16,000,000	2.654E+08	2,525	3,196
32,000,000	5.530E+08	5,254	6,639
64,000,000	1.150E+09	10,840	13,902
128,000,000	2.389E+09	22,363	28,736
256,000,000	4.956E+09	46,263	58,474